The Problems of Learning a Lexicon with a Formal Grammar

by

Sofia Hörmander

SICS R88019
Research Report
ISSN 0283-3638

The Problems of Learning a Lexicon with a Formal Grammar

Sofia Hörmander

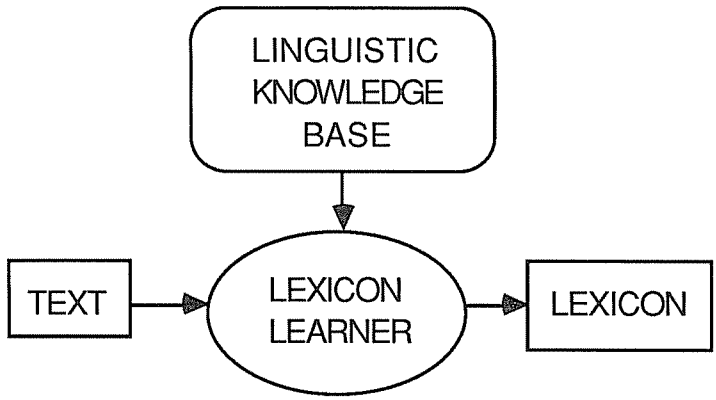
Swedish Institute of Computer Science
Box 1263
S-164 28 KISTA
Sweden
Tel: +46-8-7521500

Abstract

The possibility of using a logic grammar as the basis for automatically generating a part-of-speech lexicon is investigated, and an implementation of such a lexicon learner is demonstrated together with several possible improvements concerning efficiency and other factors.

Acknowledgements

I would like to thank Åsa Hugosson, Manny Rayner and Dag Wedelin for their help and the many useful comments they have given to me during my work. I would also like to thank Eva Enderlein for her assistance in editing.



The lexicon learner

1. Introduction

In this paper I intend to give an overview of the present state of the lexicon learning project at SICS. The aim of this project is to implement a procedure capable of inferring a part-of-speech lexicon from sample sentences with the help of a logic grammar; the original idea being that there should exist only a restricted number of lexicons consistent with the assumption that the sentences are grammatical, which in this case means that they can be generated by the grammar. My work represents a continuation of that described in Rayner et al. (1988), though with a different approach.

As far as the program is concerned the major differences between the old and the new versions are shown in figure 1.

	Knowledge Base	Method
Old	Prolog grammar	One processing of the text Several lexicons - one for every possible combination of pairs Word - Part of speech
New	Prolog grammar All words from closed categories Morphological information	Several processings of the text One lexicon only, expressing all possibilities

Figure 1. The main differences between the old and new implementations

My first step was to augment the knowledge base of the system, i. e. to supply it with more linguistic information than a prolog grammar. This reduced the risk of faulty analyses and led to a large gain in efficiency. Still, the system was unacceptably slow on all but the simplest and shortest text samples. Moreover, it could not be run on larger and/or grammatically more complex samples since overflow occurred. This was because the system, throughout the processing of a text, represented every possible combination of pairs Word - Part(s) of speech as a separate lexicon, which quickly led to a combinatorial explosion. Apart from the major problem of overflow, each sentence had to be parsed in every possible way in relation to all the lexicons produced so far (easily several thousand)--an unacceptable situation especially since most of these lexicons were the same as far as the words of the new sentence were concerned.

Because of this I chose a different representation in which a single lexicon summarized the different possibilities for every word. The price you have to pay for this is that the sentences have to be processed several times over (except if they are unambiguous from the start, of course). The new system is also allowed to ask questions about a word when it can't get any further with the knowledge it already has.

These modifications of the original method constitute one step on the way to the ultimate goal, i. e. to be able to run the system on large samples of real text that need a large grammar and where words do not necessarily recur in different syntactic contexts, often not at all. To achieve a practically working system, it will be necessary to improve it further, mainly through the incorporation of morphological information to a larger extent and a different parsing algorithm, possibly a combined top-down bottom-up parser (see sections 6 and 7).

2. The Knowledge Base

The lexicon learner has access to three kinds of information:

i) a logic grammar, at present an XG (extraposition grammar, see Pereira (1981)),

ii) a lexicon with the words from closed categories--pronouns, prepositions, conjunctions, articles and auxiliary verbs--and

iii) morphological information about the words of the language.

The idea behind giving the system words from closed categories is that these parts of speech usually remain stable--languages rarely, if ever, invent or borrow a new conjunction, for instance, and such words are never domain dependent. If a word belonging to one of these classes also can belong to one or several others, this information is given too. This leads to a considerable simplification and, most importantly, to an enormous reduction of the search space. From a "human" point of view, too, this method is preferable. Consider the problem of finding the correct part of speech assigments in the following paragraph (the words have been replaced with numbers to simulate the system´s previous total lack of knowledge).

1 2 3 4 5 6 7
8 9 4 10 11 5 12
13 14 15 5 16
4 5 17 18 19.

Most people, I believe, would find this rather difficult (though perhaps not impossible--cryptographers might find it an interesting exercise). Naturally this sample is especially difficult since it is so short, but it might still serve to illustrate the ideas. With the words belonging to closed categories given from the start the paragraph is transformed to :

It was 1 and the 2 3
did 4 and 5 in the 6
all 7 were the 8
and the 9 10 11.

The problem of finding the correct parts of speech for the numbers (words) is

now in certain cases trivial--6, for instance, is bound to be a noun.

A further improvement of performance is achieved by giving the system access to some morphological information. The effect of supplying the lexicon learner with full information can be illustrated by the paragraph from above, with all "words" given.

'Twas brillig and the slithy toves
did gyre and gimble in the wabe
all mimsy were the borogoves
and the mome raths outgrabe. (Carroll 1985)

With complete information about the closed categories and the morphology of English, the task of finding the correct lexicon for all the words is now easy--at least for a human. The only difficult word seems to be 'brillig'. Is it a noun ('It was morning') or an adjective ('It was sunny') ?

For present purposes morphological information can be regarded as belonging to two main kinds, one destructive and one constructive, see figures 2 & 3.

Part of speech ------> morphological feature(s)

eg. present participle of verb ------> X-ing

or proper noun ------> CAPITAL-X

Figure 2. Destructive morphological information

Morphological feature(s) ------> part(s) of speech

eg. X-ancy ------> noun

or X-ible ------> adjective (provided X contains at least one vowel)

Figure 3. Constructive morphological information

The learning system as it stands today uses mostly the first kind of

information, due to the difficulties of using uncertain knowledge with the present system. By using the term ´destructive´ I refer to the fact that this information is used as a filter to refute hypotheses about sentence structure that would otherwise be possible. ´Constructive´ information, on the other hand, can be used to decide the correct part of speech for a word before analyzing a text--in the few cases where morphological features are unambiguous--or else to construct a strategy for applying the rules of the grammar to a sentence. For instance, if a word has the form C*VC*able then the program could use this information in trying to parse the sentence with the word as adjective, only trying other paths (like noun - cf. constable) if this first strategy proved to be impossible or very unlikely.

If the system were able to make use of uncertain morphological information it could also abandon its present simplistic assumption that different strings represent different words. For example, if a text contains the three words X, X-ing and X-s, it is reasonable to assume that they are different forms of one and the same word, i. e. a verb (though not necessarily!). With only X and X-s given, to give a more uncertain example, there are three possibilities:

i) a noun in the singular and the plural,

ii) a verb in the infinitive or in a finite form (not 3 sing. present) and in the third person singular present, respectively, or

iii) that the two word forms do not represent the same word. In most cases, however, i) or ii) will be the case and so the system could simply start with the assumption that they represent the same word, no matter which part of speech it will turn out to be (noun or verb). Both word forms, though, might also belong to other parts of speech, a possibility which may not be excluded.

3. The Method

When given a text, the system starts by building an empty lexicon containing the words that do not belong to closed categories. It then parses the sentences in the input text one by one, several times, until all the words have been

unambiguously assigned a (list of) part(s) of speech. This works as follows (an outline of the code is given in figure 4 and a very simple example is given for illustration in figure 5). For every sentence

i) the system first builds one lexicon (with only the words of the sentence) for every possible parse (with removal of duplicates lexicons, if any). A possible parse is defined to be a parse where the words are assigned entries consistent with the lexicon produced by the previous sentences, subsequently to be called the mother lexicon. This means that the system tries to avoid, as far as possible, assigning additional entries to the words in the input sentence for which part-of-speech hypotheses already exist (see section 4).

ii) It then merges these lexicons after examining them to see if any word has been assigned the same part of speech in all of them, in which case it marks this entry as definite. This, of course, does not mean that other assignments are impossible, since any word may well belong to several parts of speech. In one sentence, for instance, the word ´request´ may be unambiguously examined as a verb, but this must not rule out the possibility that it will be found to be a noun in some other sentence.

iii) If all the words in a sentence can be unambiguously assigned parts of speech, the sentence will not be able to contribute any further to the lexicon and will be removed so as not to slow down the following processing of the text.

iv) The "merged" sentence lexicon is inserted into the mother lexicon.

```
fill_in_lexicon([Sentence|Other_sentences],In_lex,Final_lex):-
%-------------------------------------------------------------------------------------------------------
% The mother lexicon is divided into one special lexicon, containing only the words
% of the sentence, and one lexicon with the reduced mother lexicon (this is
% done for efficiency reasons). The special lexicon is used to constrain the possible
% parses of a sentence. The empty lexicon also produced will be used during parsing
% to fill in part of speech hypotheses.
%-------------------------------------------------------------------------------------------------------
  divide_lex(In_lex,Sentence,Reduced_lex,Special_lex,Empty_lex),
%-------------------------------------------------------------------------------------------------------
% Next the actual parsing and creation of new lexicons is done (see section 4 for
% explanation of the fourth parameter (normal, one_new or two_new)).
%-------------------------------------------------------------------------------------------------------
  (fill_in_lexicon_1(Sentence,Special_lex,Empty_lex,normal,Lex_list),!;
   fill_in_lexicon_1(Sentence,Special_lex,Empty_lex,one_new(Extra),Lex_list),!;
   fill_in_lexicon_1(Sentence,Special_lex,Empty_lex,two_new(Extra1,Extra2),Lex_list)),
```

```
%-----------------------------------------------------------------------------------------------------
% Then the sentence lexicons are merged and the "definite marking"  takes place.
%-----------------------------------------------------------------------------------------------------
  compact_lex(Lex_list,Lex),
%-----------------------------------------------------------------------------------------------------
% Finally the sentence lexicons are inserted into the special lexicon, which is
% then appended to the reduced mother lexicon.
%-----------------------------------------------------------------------------------------------------
  insert_lex(Lex,Special_lex),
  append(Special_lex,Reduced_lex,New_lex),
  fill_in_lexicon(Other_sentences,New_lex,Final_lex).

% -----------------------------------------------------------------------------------------------------
% -----------------------------------------------------------------------------------------------------
fill_in_lexicon_1(Sent,In_lex,L,Parse_type,Lex_list) :-
                                    findall(L,parse(Sent,In_lex,L,Parse_type),Unsorted_group),
  remove_duplicates(Unsorted_group,Lex_list).

% -----------------------------------------------------------------------------------------------------
% -----------------------------------------------------------------------------------------------------
 parse(Sent,Lex,L,How) :-
    s(Sent,[],[],[],lex(How,Lex,L)).
```

Figure 4. An outline of the main procedure of the lexicon learner

This continues until the whole text has been analysed once. Then, so as to get rid of hypotheses that might prove to be wrong, all entries that have not been marked as definite are removed before the text is reprocessed. The system continues in this way until no more combinations Word - Part of speech can be marked as definite.

```
| ?- test_group(5).
%-----------------------------------------------------------------------------------------------------
% Group 5 is a group of 6 sentences, numbered 1,2,3,4,5 and 26.
%-----------------------------------------------------------------------------------------------------
Starting run. 6 sentences, 6 words to learn.
%-----------------------------------------------------------------------------------------------------
% During the processing of the text, every sentence is printed, together with
% i) a number indicating how many different lexicons were produced for the
% sentence (most often there are  many possibilities to parse the sentences,
% but in this sample they are all unambiguous from the start)
% ii) the lexicon produced for the sentence (for illustration purposes only)
%-----------------------------------------------------------------------------------------------------

1. the cat saw the dog  1

cat: noun
dog: noun
saw: verb(obj,finite)
```

```
2. the dog saw a cat  1

cat: noun
dog: noun
saw: verb(obj,finite)

 26. that man saw the dog  1

dog: noun
saw: verb(obj,finite)
man: noun

3. a man saw the nice dog  1

dog: noun
saw: verb(obj,finite)
man: noun
nice: adj

4. the nice dog likes the man  1

dog: noun
man: noun
nice: adj
likes: verb(obj,finite)

5. the man likes the dog that the cat saw  1

dog: noun
man: noun
likes: verb(obj,finite)
saw: verb(obj,finite)
cat: noun

%-----------------------------------------------------------------------------------------------
% Finally the result is printed
%-----------------------------------------------------------------------------------------------

Run time = 1280.
cat: noun
dog: noun
likes: verb(obj,finite)
man: noun
nice: adj
saw: verb(obj,finite)

yes
```

Figure 5. A very simple example

Unfortunately everything cannot be discovered this way, simply because most texts contain quite a number of words, that occur seldom, and/or in similar,

syntactically ambiguous, contexts. Therefore the system stops at intervals to ask questions about ambiguous words, starting with those occurring many times,i. e. with those which might contribute the most to the analysis of the other ambiguous words. This whole process is then repeated until no more ambiguities remain. To further illustrate the workings of the lexicon learner there are two example runs of a simple text in appendices 1 & 2, the first with a very small and the second with a much enlarged grammar (the grammars, called 0 and 3, are summarized in appendix 3).

4. The Problem

The major difficulty when implementing a lexicon learner is that a single occurrence of a word usually does not provide enough unambiguous information to find the correct part of speech. There are two basic ways of dealing with this. One is to consider only single occurrences of words with the uncertain information they supply--this is the usual statistical approach (see for example Atwell et al. (1984) or Brodda (1983)), often used for tagging purposes. The other, which is the one chosen in this project, is to use information from several occurences of the same word form--but that leads to a new Problem: since words often belong to several parts of speech, it is then no longer possible to know if one is dealing with one and the "same" word or not.

The extent to which words will belong to several parts of speech naturally depends on how many classes you choose to have, but as soon as you have more than one, it is bound to happen. It is therefore impossible to constrain the system by only allowing one entry per word form. The opposite method, allowing any number of entries, leads to an explosion of the number of possibilities and therefore gets you nowhere. In short, you have to introduce some constraint on the number of entries a word can get, something which has proved to be very difficult to do in a non arbitrary way.

The present heuristic of the system is simply to keep the number of different entries per word as low as possible. That is, if it is possible to parse a sentence in

one or more ways without assigning any additional parts of speech to the words that have been analyzed earlier, it is done. Otherwise, the system tries to assign a new part of speech to one of the words only, or else to two if that is not enough. Apart from the problem that this method, in the worst case, might lead to the same work being done three times (first the parser tries all possible parses with the no new entry assumption, then with the one new entry ditto etc.), it might also lead to an incomplete, or worse, incorrect result. This unfortunate state of affairs arises from the fact that the system sometimes sticks to a faulty (at least as far as the new occurence is concerned) analysis of a word longer than it should.

Consider the following text (made up for the occasion), where the word ´asked´ occurs only twice :

...

They asked me.

...

He asked me if I wanted something.

...

The first sentence will supply ´asked´ as a transitive verb. Provided the grammar contains rules covering sentences like "You can come to my house if you want to have dinner.", the possibility that ´asked´ is also a verb with object and indirect question complement will not occur to the system on finding the second sentence, and so the result will be incomplete.

Worse, a grammatical sentence might be wrongly analyzed due to the heuristic of the system of minimizing the number of entries per word and so give rise to truly faulty assignments that will then haunt the parsing of the rest of the text sample. There is also the nasty possibility that an ungrammatical sentence might pass (cf. section 5) with the same consequences. My hopes are that most of these problems might be solved with a different parsing strategy, see section 7.

5. Grammar Problems

Besides the major problem that words often belong to several parts of speech, there are of course other problems that arise in connection with an implementation. In a way, though, most of them are connected with one another and with the Problem and might therefore find their solution together (see section 7).

The grammar overgenerates vastly. "All grammars leak" but here this problem manifests itself to an even higher degree than usual, since it is not possible for the grammar rules to make use of any semantic or word specific information whatsoever. The two sentences

He bakes good bread.
He left last night.

do not have the same syntactic structure, for instance, but the system has no possibility to detect this. Both sentences will give rise to two possibilities for the verb: intransitive or transitive. Hopefully, the sample will also contain other sentences which will disambiguate the words, but this is naturally not always the case.

Grammar no	Number of rules	Runtime in sec	Number of questions asked
0	44	14	1
1	56	57	2
2	77	172	2
3	104	141	4

Figure 6. Four test runs with test_group(0) (cf appendix 1; a summary of the differences between the grammars can be found in appendix 2)

In general, two problems arise when the grammar is enlarged, see figure 6. First, the time needed to parse a text is much increased, mainly because of the pure top down parsing algorithm used at present (see section 6). Secondly, and worse, the performance of the system may be severely impaired. Not all rules have this effect--often for the simple reason that the words from closed categories can tell whether the new rule is applicable or not--but once in a while this problem makes itself very noticeable. I had, for example, a problem with noun-noun modifications, as in the phrase ´snow pants´. After I included a rule for these, the system was rarely capable of finding adjectives without help (not surprisingly, since the differences between these two parts of speech are to a large extent of a semantic nature (and morphological, though this is not very prominent in English)). This effect might be avoided to a certain degree simply by giving the system larger chunks of text, but the fact remains that there are few positions where a word is unambiguously an adjective. Consider the following sentences for illustration :

Most cars are white.
Most cars are junk.
Little Bear got a pair of snow pants.
Little Bear got a pair of blue jeans. (?!)

(I have used a version of Little Bear as a test text, see appendix 4)

Giving more morphological information to the system will also help, but what is really needed is a different parsing method (see section 7).

In this specific instance another possibility would be to skip the distinction and use the old classification with nomina comprising both nouns and adjectives, but such a solution might be undesirable for several reasons. First, there is a syntactic difference between the two. `He is a tall´ is usually considered ungrammatical, though not always (Keenan & Faltz 1985). Secondly, this might lead to other words being analyzed incorrectly, at least if they occur only a few

times. An example of this can be seen in appendix 2--where a grammar is used that includes noun-noun modifications. There the system can no longer find the correct assignment for ´today´, which occurs only in the construction "Name Transitive-verb Det Noun today", without help.

The last problem I want to deal with in this section concerns ungrammatical sentences, which, in the worst case, might avoid detection. In the present system there are two possible ways of finding these sentences. In the first place there are the words belonging to closed categories; they help a lot. Many ungrammatical sentences will simply not be possible to parse, since these words may never be assigned new parts of speech. The other mechanism for detecting them is the constraint that no more than two words in a sentence may be assigned new parts of speech (cf. section 4). It is, though, possible that a faulty sentence will jump these two fences and so give rise to troublesome assignments (this might also happen with grammatical sentences being parsed incorrectly, see section 4).

6. The Problem of Efficiency

Apart from the theoretical problems considered earlier there is the very practical problem of efficiency. The lexicon learner, as described earlier, uses an XG, which is executed top down. This grammar formalism was originally chosen because it is clean, and therefore easy to write and use. Unfortunately pure top down parsing is not very well suited--except for initial experiments--for a lexicon learner. This is because the system has to find every possible parse of every sentence--only having definite information about perhaps 30-40% of the words in the sentence. This means an enormous amount of backtracking--many paths through the grammar will prove useless (and this will often occur later than if you work with a complete lexicon), and even when there is success the system has to backtrack to check all other possibilities. For long sentences some 20-30 different successful parses (not considering attachment ambiguities...) is not unusual--not to mention all the unsuccessful ones tried! My Little Bear text, for instance, (to be found in appendix 4) needed *days* to be analyzed...

Therefore the problem of having a lot of work being done over and over again becomes much more embarrassing than usual with pure top down parsing using backtracking. Consider the following situation where the next sentence to be parsed is

There were a lot of children playing games and having cake and soda pop in the kitchen.

If we suppose, for the sake of the example, that none of the words in the sentence have been encountered previously, there exists a very large number of different parses of it. All of them, though, will agree on making ´kitchen´ a noun, and ´in the kitchen´ a PP. Unaware of this the system will every time around try all possibilities for ´in´--is this an adverb?, is it a conjunction? etc., etc.--the same for ´the´ and then have to find out again that ´kitchen´ is, in fact, a noun. The time used for all these futile search paths becomes quite considerable after a while, and so it would be highly preferable only to try them once.

Because of the unusual working conditions of the parser of the lexicon learner it will be necessary, to achieve a practically working system, to find another formalism which combines top down and bottom up techniques and where partial results, like complete phrases, can be saved when the system backtracks (or else one where the system does not have to backtrack over correct choices, i. e. where some kind of island parsing technique is used). A "left corner" parser (i. e. a kind of bottom up strategy with top down filtering, see Alshawi et al. (1987)) might be a useful step in this direction, if modified to take the somewhat special bottom up information of the lexicon learner into consideration (only some words given and then uncertain lexical and perhaps morphological information about the others, cf. section 7). Probably, though, what is really desirable is a new grammar formalism tailored for the special needs of a lexicon learner. The next section will hopefully provide some insight into how this might look.

7. Further Improvements

We have seen above how the Problem and other problems can make life difficult for a lexicon learner of the sort I have been sketching in this paper. What is needed is not only to make the parser work more efficiently (see previous section) but also to use the work of the parser in a more efficient and theoretically pleasing way. To achieve this the system needs more knowledge about the language, a parsing algorithm capable of using this additional knowledge in a constructive way and, of course, an overall implementation modified accordingly.

The grammar of the lexicon learner today considers every parse of a sentence as equally plausible. If there is one single parse of a sentence with the no new entry assumption (see section 4) then this is the path the system will choose, no matter how unusual a construction this is, or how morphologically unlikely it may be.

Instead of this very simple rule saying that the most plausible analyses of a sentence are those that are most consistent with the mother lexicon what is needed is a refined ´plausibility rule´ considering also other factors. The possible extensions to the knowledge base I have in mind are then :

i) More constructive morphological information, as well as information about the morphology of inflected forms of words (see section 2)--to be used dynamically by the parser (which then cannot work purely top down) to reorder alternative grammar rules.

ii) Information about the frequency of applicability of grammar rules. To take an example, if a sentence contains the phrase "blue jeans" (cf. section 5), and if this is the first time the word ´blue´ occurs, then the possibility of ´blue´ being a noun should not be explored in the first place. The more complete the grammar the more this kind of information becomes necessary.

iii) Information about ´co-occurence compatibilities´ of different parts of speech. Such knowledge, just as i) and ii), is not absolute. Probably no two parts

of speech are completely incompatible, if you consider the case of homographs. Some combinations, though, are very unusual, whereas others occur frequently. In the present system I distinguish between nouns, adjectives, three kinds of adverbs, and verbs, and among these I discriminate between verbs with different kinds of complements and with different forms (finite, present and perfect participle, and infinitive). With such a classification it is clear that a lexicon learner should be much more reluctant to assign to a word, say, ´adverb and noun´ than ´finite form of a transitive verb and infinitive of a transitive verb´ (to take a particularly obvious example).

Incorporating all of this should lead to a better performance of the system not only in terms of efficiency but also in terms of the correctness and completeness of the output. It will be a challenge for future work with the system to integrate more and more of this knowledge and to define interactions between the different knowledge domains that are theoretically well motivated.

8. Bibliography

Alshawi, H., R.C. Moore, D.B. Moran and S.G. Pullman (1987): *Research Programme in Natural Language Processing,* Annual Report for SRI Project 2989, SRI International.

Atwell, E.S., G.N. Leech and R.G. Garside (1984): *Analysis of the LOB Corpus: progress and prospects,* in J. Aarts and W. Meijs, eds., Corpus Linguistics, Rodopi.

Brodda, Benny (1983): *An experiment with heuristic parsing of Swedish,* in First Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference, Pisa (ACL).

Carroll, Lewis (1985): *Through the Looking Glass,* Penguin Books Ltd.

Holmelund Minarik, Else (1957): *Little Bear,* Harper & Row, Publishers, New York.

Keenan, Edward L. and Leonard M. Faltz (1985): *Boolean Semantics for Natural Language,* D. Reidel Publishing Company.

Pereira, Fernando (1981): *Extraposition Grammars,* in American Journal of Computational Linguistics, Vol.7, 243-256.

Rayner, Manny, Åsa Hugosson and Göran Hagert (1988): *Using a Logic Grammar to Learn a Lexicon,* SICS Research Report.

Appendix 1

```
SICStus 0.6 #4EXPERIMENTAL: Mon Jun 13 12:41:34 MET DST 1988
Copyright (C) 1987, Swedish Institute of Computer Science.
All rights reserved.
| ?- ['st.pl'].
{consulting /khons/asa/L2/st.pl...}
{compiling /khons/asa/L2/xgproc.pl...}
{xgproc compiled, 8459 msec 15250 bytes}
{consulting /khons/asa/L2/xgrun.pl...}
{xgrun consulted, 140 msec 774 bytes}
{compiling /khons/asa/L2/top-1.pl...}
{top-1.pl compiled, 23440 msec 47903 bytes}
{consulting /khons/asa/L2/sent.pl...}
{sent.pl consulted, 3121 msec 24047 bytes}
{compiling /khons/asa/L2/parts-old.pl...}
{parts-old.pl compiled, 7500 msec 22573 bytes}
{consulting /khons/asa/L2/last.pl...}
{last.pl consulted, 500 msec 3318 bytes}

** Grammar from file so_grammar.pl : 11661 words **

{compiling /khons/asa/L2/grammar_to_compile.pl...}
{grammar_to_compile.pl compiled, 7360 msec 712 bytes}
{consulting /khons/asa/L2/read-file.pl...}
{read-file.pl consulted, 1200 msec 12780 bytes}
{st.pl consulted, 53800 msec 139507 bytes}

yes

%-------------------------------------------------------------------------
% Here I use a small grammar of about 45 rules
%-------------------------------------------------------------------------

| ?- test_group(0).

Starting run. 43 sentences, 36 words to learn.

Order before sorting:
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28
,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43]
Order after sorting:
[20,27,30,1,2,6,7,9,12,13,14,15,16,17,18,19,23,24,25,26,28,29,31,33,34,35,3
6,37,3,4,10,32,38,40,41,43,5,8,11,22,39,42,21]

%-------------------------------------------------------------------------
% The sentences are sorted according to length so as to parse the
% (possibly) least ambiguous sentences first.
% The number printed after each sentence refers to the number of different
% lexicons produced for it.
%-------------------------------------------------------------------------

20. john has not read the newspaper  1
27. the man has a cat  1
30. john ate the beans  1
1. the cat saw the dog  1
2. the dog saw a cat  1
6. most men like the dog  1
7. the dog sat on the table  2
9. the man read the newspaper  1
```

```
12. john threw the newspaper to the dog  1
13. the dog belongs to the man  2
14. the dog likes most men  1
15. the men like john  1
16. the dog sat with john  1
17. john threw the newspaper on the table  1
18. john read the newspaper today  1
19. john has read the newspaper today  1
23. the cat sat on the car  2
24. the dog can not drive the car  1
25. the man can drive the car  1
26. that man saw the dog  1
28. the man who has a cat has no dog  1
29. the cat ate a fish  1
31. the man ate a can of beans  1
33. john saw a glass on the table  1
34. john drank a glass of water  1
35. the man drank the whisky  1
36. john poured the water on the cat  1
37. john poured a can of water on the cat  1
3. a man saw the nice dog  1
4. the nice dog likes the man  1
10. the dog brought the man the newspaper  1
32. the man brought the cat a can of catfood  1
38. john hoped the dog drank the water  1
40. mary knows that john owns a dog  1
41. john believes that mary drives a car  1
43. peter can not believe that mary ate the fish  1
5. the man likes the dog that the cat saw  1
8. the table belongs to the man who owns the dog  2
11. the dog hoped that the man read the newspaper  1
22. the man who owns the cat drives the car  1
39. the man hoped that john likes the dog  1
42. mary believes john knows that peter has a cat  1
21. the man read the newspaper before john saw the cat  1

 * * * * * * * * * * * * * * * *

%-----------------------------------------------------------------------
% Here the second processing of the text starts
%-----------------------------------------------------------------------

7. the dog sat on the table  1
13. the dog belongs to the man  2
23. the cat sat on the car  1
8. the table belongs to the man who owns the dog  2

 * * * * * * * * * * * * * * * *

%-----------------------------------------------------------------------
% And here the third (and so on...)
%-----------------------------------------------------------------------


belongs : verb(intrans,finite)

is this correct?
(answer y. or n.)n.

belongs : verb(pobj(to),finite)
```

```
is this correct?
(answer y. or n.)y.

 * * * * * * * * * * * * * * * *

13. the dog belongs to the man  1
8. the table belongs to the man who owns the dog  1

 * * * * * * * * * * * * * * * *


Run time = 13260.

ate: verb(obj,finite)
beans: noun
believe: verb(s_comp,inf)
believes: verb(s_comp,finite)
belongs: verb(pobj(to),finite)
brought: verb(two_objs,finite)
car: noun
cat: noun
catfood: noun
dog: noun
drank: verb(obj,finite)
drive: verb(obj,inf)
drives: verb(obj,finite)
fish: noun
glass: noun
hoped: verb(s_comp,finite)
john: name
knows: verb(s_comp,finite)
like: verb(obj,finite)
likes: verb(obj,finite)
man: noun
mary: name
men: noun
newspaper: noun
nice: adj
owns: verb(obj,finite)
peter: name
poured: verb(obj,finite)
read: verb(obj,perf_part) verb(obj,finite)
sat: verb(intrans,finite)
saw: verb(obj,finite)
table: noun
threw: verb(obj,finite)
today: adv
water: noun
whisky: noun
yes
| ?- ^D
{ End of SICStus execution, user time 67.360 }
```

Appendix 2

```
> sicstus.test
SICStus 0.6 #4EXPERIMENTAL: Mon Jun 13 12:41:34 MET DST 1988
Copyright (C) 1987, Swedish Institute of Computer Science.
All rights reserved.
| ?- ['st3.pl'].
{consulting /khons/asa/L2/st3.pl...}
{compiling /khons/asa/L2/xgproc.pl...}
{xgproc compiled, 8560 msec 15250 bytes}
{consulting /khons/asa/L2/xgrun.pl...}
{xgrun consulted, 160 msec 774 bytes}
{compiling /khons/asa/L2/top-1.pl...}
{top-1.pl compiled, 23500 msec 47903 bytes}
{consulting /khons/asa/L2/sent.pl...}
{sent.pl consulted, 3120 msec 24047 bytes}
{compiling /khons/asa/L2/parts3.pl...}
{parts3.pl compiled, 7960 msec 23355 bytes}
{compiling /khons/asa/L2/last.pl...}
{last.pl compiled, 1000 msec 4362 bytes}

** Grammar from file so_grammar-3.pl : 24768 words **

{compiling /khons/asa/L2/grammar_to_compile.pl...}
{grammar_to_compile.pl compiled, 16500 msec 830 bytes}
{consulting /khons/asa/L2/read-file.pl...}
{read-file.pl consulted, 1280 msec 12780 bytes}
{st3.pl consulted, 66400 msec 154555 bytes}

yes

%-------------------------------------------------------------------------
% Here I use a larger grammar of about 100 rules
%-------------------------------------------------------------------------

| ?- test_group(0).

Starting run. 43 sentences, 36 words to learn.

Order before sorting:
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28
,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43]
Order after sorting:
[20,27,30,1,2,6,7,9,12,13,14,15,16,17,18,19,23,24,25,26,28,29,31,33,34,35,3
6,37,3,4,10,32,38,40,41,43,5,8,11,22,39,42,21]

20. john has not read the newspaper  1
27. the man has a cat  1
30. john ate the beans  1
1. the cat saw the dog  1
2. the dog saw a cat  1
6. most men like the dog  1
7. the dog sat on the table  2
9. the man read the newspaper  1
12. john threw the newspaper to the dog  2
13. the dog belongs to the man  2
14. the dog likes most men  1
15. the men like john  1
16. the dog sat with john  1
17. john threw the newspaper on the table  1
```

18. john read the newspaper today 2
19. john has read the newspaper today 2
23. the cat sat on the car 2
24. the dog can not drive the car 1
25. the man can drive the car 1
26. that man saw the dog 1
28. the man who has a cat has no dog 1
29. the cat ate a fish 1
31. the man ate a can of beans 1
33. john saw a glass on the table 1
34. john drank a glass of water 2
35. the man drank the whisky 1
36. john poured the water on the cat 2
37. john poured a can of water on the cat 2
3. a man saw the nice dog 2
4. the nice dog likes the man 2
10. the dog brought the man the newspaper 1
32. the man brought the cat a can of catfood 1
38. john hoped the dog drank the water 1
40. mary knows that john owns a dog 2
41. john believes that mary drives a car 2
43. peter can not believe that mary ate the fish 1
5. the man likes the dog that the cat saw 1
8. the table belongs to the man who owns the dog 1
11. the dog hoped that the man read the newspaper 1
22. the man who owns the cat drives the car 1
39. the man hoped that john likes the dog 1
42. mary believes john knows that peter has a cat 1
21. the man read the newspaper before john saw the cat 1

* * * * * * * * * * * * * * *

7. the dog sat on the table 1
12. john threw the newspaper to the dog 1
13. the dog belongs to the man 2
18. john read the newspaper today 2
19. john has read the newspaper today 2
23. the cat sat on the car 1
34. john drank a glass of water 1
36. john poured the water on the cat 2
37. john poured a can of water on the cat 2
3. a man saw the nice dog 2
4. the nice dog likes the man 2
40. mary knows that john owns a dog 1
41. john believes that mary drives a car 1

* * * * * * * * * * * * * * *

nice : adj

is this correct?
(answer y. or n.)y.

nice : noun

is this correct?
(answer y. or n.)n.

* * * * * * * * * * * * * * * *

```
13. the dog belongs to the man  2
18. john read the newspaper today  2
19. john has read the newspaper today  2
36. john poured the water on the cat  2
37. john poured a can of water on the cat  2
3. a man saw the nice dog  1
4. the nice dog likes the man  1

 * * * * * * * * * * * * * * * *


poured : verb(obj,finite)

is this correct?
(answer y. or n.)y.

poured : verb(obj_and_pobj(on),finite)

is this correct?
(answer y. or n.)n.

 * * * * * * * * * * * * * * * *

13. the dog belongs to the man  2
18. john read the newspaper today  2
19. john has read the newspaper today  2
36. john poured the water on the cat  1
37. john poured a can of water on the cat  1

 * * * * * * * * * * * * * * * *


today : noun

is this correct?
(answer y. or n.)n.

today : sent_adv

is this correct?
(answer y. or n.)y.

 * * * * * * * * * * * * * * * *

13. the dog belongs to the man  2
18. john read the newspaper today  1
19. john has read the newspaper today  1

 * * * * * * * * * * * * * * * *


belongs : verb(intrans,finite)

is this correct?
(answer y. or n.)n.

belongs : verb(pobj(to),finite)

is this correct?
```

```
(answer y. or n.)y.

 * * * * * * * * * * * * * * * * *

13. the dog belongs to the man  1

 * * * * * * * * * * * * * * * * *


Run time = 141380.

ate: verb(obj,finite)
beans: noun
believe: verb(s_comp,inf)
believes: verb(s_comp,finite)
belongs: verb(s_comp,finite) verb(pobj(to),finite)
brought: verb(two_objs,finite)
car: noun
cat: noun
catfood: noun
dog: noun
drank: verb(obj,finite)
drive: verb(obj,inf)
drives: verb(obj,finite)
fish: noun
glass: noun
hoped: verb(s_comp,finite)
john: name
knows: verb(s_comp,finite)
like: verb(obj,finite)
likes: verb(obj,finite)
man: noun
mary: name
men: noun
newspaper: noun
nice: adj
owns: verb(obj,finite)
peter: name
poured: verb(obj,finite)
read: verb(obj,perf_part) verb(obj,finite)
sat: verb(intrans,finite)
saw: verb(obj,finite)
table: noun
threw: verb(obj,finite)
today: sent_adv
water: noun
whisky: noun
yes
| ?- ^D
{ End of SICStus execution, user time 208.180 }
```

Appendix 3

Grammar 0 covers only a very small subset of English - NP:s have to be names, pronouns or determiner + (adj) + noun + (PP) + (relative phrase). No more than one auxiliary is permitted in front of a verb, and only eight different verbal complements are admitted. Sentence modifiers are only recognized at the end of a sentence, and there may not be more than one.

Grammar 1 handles more kinds of NP, AP:s, more verbal complements, sentence modifiers of many kinds at the beginning of a sentence, and an unlimited number at the end.

Grammar 2 comprises even more rules for expanding an NP. It can handle sequences of auxiliary verbs, considerably more verbal complements, two kinds of adverbs, and, last but NOT least, coordination (this slows things up a lot).

Grammar 3 differs between NP:s in different positions in the sentence, comprises noun-noun modification, and some more verbal complements. Furthermore it accepts adjectives taking PP and S´ complements, and handles three kinds of adverbs.

Appendix 4

It was cold.
Snow was falling.
You could see the snow come down.
Little Bear told Mother Bear something.
He was cold.
Snow was falling and he wanted something to put on.
So Mother Bear made something for Little Bear.
She asked Little Bear to look and told him that she had something for her little bear.
She held it out for him.
It was a hat.
Mother Bear told Little Bear to put it on his head.
Little Bear was surprised.
He was happy and told Mother Bear that now he would not be cold.
Little Bear went out to play.
Then Little Bear comes in again.
Mother Bear asked him if he wanted something.
Little Bear told Mother Bear that he was cold.
He wanted something to put on.
So Mother Bear made something for Little Bear.
She asked Little Bear to look and told him that she had something for her little bear.
Mother Bear told Little Bear to put it on.
Little Bear was surprised.
It was a coat.
He was happy and told Mother Bear that now he would not be cold.
Little Bear went out to play.
Then Little Bear comes in again.
Mother Bear asked him if he wanted something.
Little Bear told Mother Bear that he was cold.
He wanted something to put on.
So Mother Bear made something for Little Bear again.
She asked Little Bear to look and told him that she had something for her little bear.
She told him that now he could not be cold and asked him to put it on.
Little Bear was surprised.
It was a pair of snow pants.
He was happy and told Mother Bear that now he would not be cold.
Little Bear went out to play.
Then Little Bear comes in again.
Mother Bear is worried.
She wonders what he can want now.
Little Bear tells he that he is cold.
He wants something to put on.
Mother Bear thinks that her little bear has a coat and a hat and a pair of

pants.
She asks him if he would like a fur coat too.
Little Bear was happy.
He wanted a fur coat too.
So Mother Bear took the hat and the coat and the pants and then she said to him that the new coat was right there.
Little Bear was happy and said that now he was sure that he would not be cold any more.
He was not cold either.
This was a wonderful story.
I hope you think so too.